Including vector graphics in HTML

- What are vector graphics?
- What is SVG?
- Adding SVG to your pages
  - The quick way: img element
    - Pros
    - Cons
  - Troubleshooting and cross-browser support
  - How to include SVG code inside your HTML
    - Pros
    - Cons
  - How to embed an SVG with an iframe
    - Cons
- Playing with SVG
- Summary
- See also

Vector graphics are very useful in many circumstances — they have small file sizes and are highly scalable, so they don't pixelate when zoomed in or blown up to a large size. In this article we'll show you how to include one in your webpage.

Prerequisites:	You should know the basics of HTML and how to insert an image into your document.
Objective:	Learn how to embed an SVG (vector) image into a webpage.

> **NOTE:** This article doesn't intend to teach you SVG; just what it is, and how to add it to web pages.

What are vector graphics?

On the web, you'll work with two types of images — **raster images**, and **vector images**:

- **Raster images** are defined using a grid of pixels — a raster image file contains information showing exactly where each pixel is to be placed, and exactly what color it should be. Popular web raster formats include Bitmap (`.bmp`), PNG (`.png`), JPEG (`.jpg`), and GIF (`.gif`).
- **Vector images** are defined using algorithms — a vector image file contains shape and path definitions that the computer can use to work out what the image should look like when rendered on the screen. The SVG format allows us to create powerful vector graphics for use on the Web.

To give you an idea of the difference between the two, let's look at an example:

```
<img src="star.png" alt="A raster star" />
<img src="star.svg" role="img" alt="A vector star" />
```

This shows two seemingly identical red stars with black drop shadows, side by side. The difference is that the left one is a raster (PNG), and the right one is a vector (SVG) image.

The difference becomes apparent when you zoom in on the page or increase the size of the images. The following shows how the two stars render at a width of `300px`:

```
img {
  width: 300px;
}
```

The PNG image becomes pixelated because it contains information on where each pixel should be (and what color). When it is zoomed, each pixel is increased in size to fill multiple pixels on screen, so the image starts to look blocky. The SVG image however continues to look nice and crisp, because no matter what size it is, the algorithms are used to work out the shapes in the image, with the values being scaled as it gets bigger.

Moreover, vector image files are much lighter than their raster equivalents, because they only need to hold a handful of algorithms, rather than information on every pixel in the image individually.

What is SVG?

SVG is an XML-based language for describing vector images. It's basically markup, like HTML, except that you've got many different elements for defining the shapes you want to appear in your image, and the effects you want to apply to those shapes. SVG is for marking up graphics, not content. SVG defines elements for creating basic shapes, like circle and rect, as well as elements for creating more complex shapes, like path and polygon. More advanced SVG features include feColorMatrix (transform colors using a transformation matrix), animate (animate parts of your vector graphic), and mask (apply a mask over the top of your image).

As a basic example, the following code creates a circle and a rectangle:

```
<svg
  version="1.1"
  baseProfile="full"
  width="300"
  height="200"
  xmlns="http://www.w3.org/2000/svg">
  <rect width="100%" height="100%" fill="black" />
  <circle cx="150" cy="100" r="90" fill="blue" />
</svg>
```

This creates the following output:

From the example above, you may get the impression that SVG is easy to hand-code. Yes, you can hand-code simple SVG in a text editor, but for a complex image this quickly starts to get very difficult. For creating SVG images, most people use a vector graphics editor like Inkscape or Illustrator. These packages allow you to create a variety of illustrations using various graphics tools, and create approximations of photos (for example Inkscape's Trace Bitmap feature.)

SVG has some additional advantages besides those described so far:

- Text in vector images remains accessible (which also benefits your SEO).
- SVGs lend themselves well to styling/scripting, because each component of the image is an element that can be styled via CSS or scripted via JavaScript.

So why would anyone want to use raster graphics over SVG? Well, SVG does have some disadvantages:

- SVG can get complicated very quickly, meaning that file sizes can grow; complex SVGs can also take significant processing time in the browser.

- SVG can be harder to create than raster images, depending on what kind of image you are trying to create.

Raster graphics are arguably better for complex precision images such as photos, for the reasons described above.

SVG graphics exported from editors such as Inkscape have great room for size optimization. Before deploying them to the web, you probably want to run them through an SVG optimizer such as SVGO.

Adding SVG to your pages

In this section we'll go through the different ways in which you can add SVG vector graphics to your web pages.

The quick way: `img` element

To embed an SVG via an img element, you just need to reference it in the src attribute as you'd expect. You will need a `height` or a `width` attribute (or both if your SVG has no inherent aspect ratio). If you have not already done so, please read HTML images.

```
<img
  src="equilateral.svg"
  alt="triangle with all three sides equal"
  height="87"
  width="100" />
```

Pros

- Quick, familiar image syntax with built-in text equivalent available in the `alt` attribute.
- You can make the image into a hyperlink easily by nesting the `<img>` inside an a element.
- The SVG file can be cached by the browser, resulting in faster loading times for any page that uses the image loaded in the future.

Cons

- You cannot manipulate the image with JavaScript.
- If you want to control the SVG content with CSS, you must include inline CSS styles in your SVG code. (External stylesheets invoked from the SVG file take no effect.)

- You cannot restyle the image with CSS pseudoclasses (like `:focus`).

Troubleshooting and cross-browser support

For browsers that don't support SVG (IE 8 and below, Android 2.3 and below), you could reference a PNG or JPG from your `src` attribute and use a `srcset` attribute (which only recent browsers recognize) to reference the SVG. This being the case, only supporting browsers will load the SVG — older browsers will load the PNG instead:

```
<img
  src="equilateral.png"
  alt="triangle with equal sides"
  srcset="equilateral.svg" />
```

You can also use SVGs as CSS background images, as shown below. In the below code, older browsers will stick with the PNG that they understand, while newer browsers will load the SVG:

```
background: url("fallback.png") no-repeat center;
background-image: url("image.svg");
background-size: contain;
```

Like the `<img>` method described above, inserting SVGs using CSS background images means that the SVG can't be manipulated with JavaScript, and is also subject to the same CSS limitations.

If your SVGs aren't showing up at all, it might be because your server isn't set up properly. If that's the problem, this article will point you in the right direction.

How to include SVG code inside your HTML

You can also open up the SVG file in a text editor, copy the SVG code, and paste it into your HTML document — this is sometimes called putting your **SVG inline**, or **inlining SVG**. Make sure your SVG code snippet begins with an `<svg>` start tag and ends with an `</svg>` end tag. Here's a very simple example of what you might paste into your document:

```
<svg width="300" height="200">
  <rect width="100%" height="100%" fill="green" />
```

```
</svg>
```

Pros

- Putting your SVG inline saves an HTTP request, and therefore can reduce your loading time a bit.
- You can assign `class`es and `id`s to SVG elements and style them with CSS, either within the SVG or wherever you put the CSS style rules for your HTML document. In fact, you can use any SVG presentation attribute as a CSS property.
- Inlining SVG is the only approach that lets you use CSS interactions (like `:focus`) and CSS animations on your SVG image (even in your regular stylesheet.)
- You can make SVG markup into a hyperlink by wrapping it in an a element.

Cons

- This method is only suitable if you're using the SVG in only one place. Duplication makes for resource-intensive maintenance.
- Extra SVG code increases the size of your HTML file.
- The browser cannot cache inline SVG as it would cache regular image assets, so pages that include the image will not load faster after the first page containing the image is loaded.
- You may include fallback in a foreignObject element, but browsers that support SVG still download any fallback images. You need to weigh whether the extra overhead is really worthwhile, just to support obsolescent browsers.

How to embed an SVG with an `iframe`

You can open SVG images in your browser just like webpages. So embedding an SVG document with an `<iframe>` is done just like we studied in From <object> to <iframe> — general embedding technologies.

Here's a quick review:

```
<iframe src="triangle.svg" width="500" height="500" sandbox></iframe>
```

This is definitely not the best method to choose:

Cons

- `<iframe>` elements can include fallback content between their opening and closing tags, but this is only displayed in browsers that don't support `<iframe>`s, not when the image fails to load.

- Moreover, unless the SVG and your current webpage have the same origin, you cannot use JavaScript on your main webpage to manipulate the SVG.

Playing with SVG

In this exercise we'd like you to have a go at playing with some SVG. Press the **Play** button to open the next example in the MDN Playground and edit it there.

Go to the SVG Element Reference to see what other elements you can use that bring a lot of built-in functionality. There are other shapes you can try, like ellipses, or you can experiment with patterns, or even filter effects. This section is all about your research skills, trying something new, and having some fun.

If you get stuck and can't get your code working, you can always reset it using the *Reset* button in the Playground.

```
<svg width="100%" height="100%">
  <rect width="100%" height="100%" fill="red" />
  <circle cx="100%" cy="100%" r="150" fill="blue" stroke="black" />
  <polygon points="120,0 240,225 0,225" fill="green" />
  <text
    x="50"
    y="100"
    font-family="Verdana"
    font-size="55"
    fill="white"
    stroke="black"
    stroke-width="2">
    Hello!
  </text>
</svg>
```

Summary

This article has provided you with a quick tour of what vector graphics and SVG are, why they are useful to know about, and how to include SVG inside your webpages. It was never intended to be a full guide to learning SVG, just a pointer so you know what SVG is if you meet it in your travels

around the Web. So don't worry if you don't feel like you are an SVG expert yet. We've included some links below that might help you if you wish to go and find out more about how it works.

See also

- SVG tutorial on MDN
- Sara Soueidan's tutorial on responsive SVG images
- SVG Properties and CSS
- How to scale SVGs (it's not as simple as raster graphics!)

□□□ #3
□ □□□ □□□ 2026-10-31 03:27:53 UTC
□ cocobook-sync □□□ 2026-11-04 17:32:50 UTC